

Spanning Trees in 2-trees

P. Renjith*, N. Sadagopan*, Douglas B. West[†]

July 21, 2016

Abstract

A spanning tree of a graph G is a connected acyclic spanning subgraph of G . We consider enumeration of spanning trees when G is a 2-tree, meaning that G is obtained from one edge by iteratively adding a vertex whose neighborhood consists of two adjacent vertices. We use this construction order both to inductively list the spanning trees without repetition and to give bounds on the number of them. We determine the n -vertex 2-trees having the most and the fewest spanning trees. The 2-tree with the fewest is unique; it has $n-2$ vertices of degree 2 and has $n2^{n-3}$ spanning trees. Those with the most are all those having exactly two vertices of degree 2, and their number of spanning trees is the Fibonacci number F_{2n-2} .

1 Introduction

The problem of counting spanning trees in a graph has attracted much attention [1, 2, 3, 5, 6, 7, 8, 9]. This number can be computed efficiently using the Matrix Tree Theorem [4]. By “enumeration” we instead mean the listing of all spanning trees. We consider the special case of 2-trees, which form a well-known subclass of the family of chordal graphs. A 2-tree is a graph obtained from the graph consisting of two adjacent vertices by iteratively adding one new vertex whose neighborhood consists of two adjacent vertices.

For general graphs, the tree-listing algorithms in the literature use backtracking or generate trees from fundamental cycles [1, 6]. For n -vertex 2-trees, we show that the number of spanning trees is always between 2^{n-2} and 3^{n-2} , so it is not possible to list them in polynomial time. Nevertheless, the structural description of 2-trees yields a simple sequential algorithm that lists all spanning trees without repetition. This approach is simpler than that in [1, 6], but the domain of 2-trees is much simpler than the contexts of those papers. Given a vertex v of degree 2 in a 2-tree G , we list the spanning trees in G in terms of the spanning trees in $G - v$, separately those in which v is a leaf and those in which v is not a leaf.

¹Indian Institute of Information Technology, Design, and Manufacturing, Kancheepuram, Chennai, India.

²Departments of Mathematics, Zhejiang Normal University, Jinhua, Zhejiang, China, and University of Illinois, Urbana, Illinois, USA. Research supported by Recruitment Program of Foreign Experts, 1000 Talent Plan, State Administration of Foreign Experts Affairs, China.

This algorithm, presented in Section 2, immediately yields lower and upper bounds 2^{n-2} and 3^{n-2} on the number of spanning trees in an n -vertex 2-tree. We improve both bounds to obtain optimal values and characterize the n -vertex 2-trees achieving equality. We provide the constructions in Section 3 and prove optimality in Sections 4 and 5.

The minimum number of spanning trees in an n -vertex 2-tree is $n2^{n-3}$, achieved uniquely by the 2-tree having $n - 2$ vertices of degree 2. This 2-tree is obtained from the graph with two vertices and one edge by iteratively adding vertices of degree 2 adjacent to the two original vertices; it is sometimes called the n -book, denoted B_n .

The n -vertex 2-trees having the most spanning trees are those having exactly two vertices of degree 2. Somewhat surprisingly, all such 2-trees have the same number of spanning trees. The value is asymptotic to $\frac{1}{\sqrt{5}}(2.618)^{n-2}$; the exact value is the Fibonacci number F_{2n-2} , where $F_0 = 0$, $F_1 = 1$, and $F_m = F_{m-1} + F_{m-2}$ for $m \geq 2$. Such 2-trees include the graphs obtained from an n -vertex path by (1) making one endpoint adjacent to all the other vertices or (2) adding edges joining vertices separated by distance 2 along the path.

We use standard graph-theoretic notation. In particular, $V(G)$ and $E(G)$ denote the vertex set and edge set of a graph G . The *neighborhood* of a vertex v is the set $N(v)$ of vertices adjacent to v , that is, the set $\{u \in V(G) : uv \in E(G)\}$. The *complete graph* K_n is the n -vertex graph in which any two vertices are adjacent, and a *clique* is a set of pairwise adjacent vertices. The subgraph of a graph G *induced* by a vertex subset S , written $G[S]$, is the subgraph defined by $V(G[S]) = S$ and $E(G[S]) = \{uv \in E(G) : u, v \in S\}$. When $v \in V(G)$, we write $G - v$ for the graph $G[V(G) - \{v\}]$.

A vertex in a graph is *simplicial* if its neighborhood is a clique. A *simplicial elimination ordering* of a graph G is an ordering (v_n, \dots, v_1) of $V(G)$ such that each v_i is a simplicial vertex of the induced subgraph $G[\{v_i, \dots, v_1\}]$. Traditionally, simplicial elimination orderings have also been called “perfect elimination orderings” (see [11]), but there are now many types of elimination orderings, so using the word “simplicial” is more informative. Our definition of 2-trees explicitly constructs them in the reverse of a simplicial elimination ordering.

A graph is *chordal* if no induced subgraph is a cycle of length at least 4. It was proved by Dirac [12] that a graph is chordal if and only if it has a simplicial elimination ordering. Thus the 2-trees form a special class of chordal graphs. They are the connected chordal graphs having a simplicial elimination ordering in which except for the last two vertices, all vertices have degree 2 when deleted. We thus call a simplicial elimination ordering of a 2-tree a *2-simplicial ordering*. Note that 2-trees have no cut-vertices, and hence a vertex in a 2-tree is simplicial if and only if it has degree 2.

2 Enumeration without Repetition

Given a 2-tree G and an associated 2-simplicial ordering (v_n, \dots, v_1) , we present an algorithm to list all spanning trees in G without repetition. Finding a 2-simplicial ordering is trivial; a 2-tree with more than two vertices has a simplicial vertex of degree 2, and every vertex of degree 2 is simplicial and can be used as the next vertex in the ordering. Hence maintaining the vertex degrees makes it very easy to generate a 2-simplicial ordering.

Given a 2-tree G with 2-simplicial ordering (v_n, \dots, v_1) , let $G_i = G[v_i, \dots, v_1]$. Starting with G_2 , which is isomorphic to K_2 , we iteratively obtain the spanning trees of G_2, \dots, G_n in order. In particular, from the spanning trees of G_{n-1} we obtain the spanning trees of G_n , by listing first those in which v_n is a leaf and then those in which v_n is not a leaf. The justification of our algorithm is based on the following observation.

Observation 2.1. Let G be a 2-tree in which v is a simplicial vertex with neighborhood $\{x, y\}$. A subgraph T of G is a spanning tree in which v is not a leaf if and only if v has degree 2 in T , the edge xy is not in T , and the spanning subgraph T' of $G - v$ with $E(T') = E(T) \cup \{xy\} - \{vx, vy\}$ is a spanning tree in $G - v$.

Algorithm 1 *Enumerate-Spanning-trees-without-repetitions(G)*

- 1: **Input:** A 2-tree G with 2-simplicial ordering (v_n, \dots, v_1) , for $n \geq 2$; define $G_i = G[\{v_i, \dots, v_1\}]$ for $1 \leq i \leq n$.
 - 2: Initialize LIST with the single spanning tree of G_2 .
 - 3: Let $\{x, y\}$ be the neighborhood of v_i among $\{v_{i-1}, \dots, v_1\}$.
 - 4: **for** each spanning tree T of G_{i-1} in LIST, **do**
 - 5: Append to LIST the two spanning trees of G_i obtained by adding the edges $v_i x$ or $v_i y$ to T .
 - 6: If $xy \in E(T)$, then Append to LIST the spanning tree of G_i obtained by replacing xy with the edges $v_i x$ and $v_i y$.
 - 7: **end for**
-

Theorem 2.2. *For a 2-tree G , Algorithm 1 lists all spanning trees of G without repetition.*

Proof. Let G have n vertices. We use induction on n , with trivial basis. Assume that the algorithm lists the spanning trees of G_{n-1} without repetition. Deleting v_n from any spanning tree of G_n where v_n is a leaf yields a spanning tree of G_{n-1} , which in the algorithm produces it in Step 5. By Observation 2.1, any spanning tree in which v_n is not a leaf is produced by the algorithm in Step 6. Thus the algorithm lists all spanning trees of G without repetition. Furthermore, since the trees are built by iteratively adding one vertex, the time taken is proportional to nt , where t is the number of spanning trees. \square

3 Bounds and Constructions

Let $T(G)$ denote the number of spanning trees in a 2-tree G . Again let $G_i = G[\{v_i, \dots, v_1\}]$, where (v_n, \dots, v_1) is a 2-simplicial ordering of G .

Theorem 3.1. *Fix $n \in \mathbb{N}$ with $n \geq 2$. If G is an n -vertex 2-tree, then $2^{n-2} \leq T(G) \leq 3^{n-2}$.*

Proof. From each spanning tree of G_{n-1} , the algorithm generates two spanning trees of G_n in which v_n is a leaf, and for each spanning tree containing the edge joining the neighbors of v_n , it generates one more tree in which v_n is not a leaf. Hence the bounds follow immediately from Theorem 2.2 by induction on n . \square

The bounds in Theorem 3.1 are not sharp. The constructions we present in this section actually minimize and maximize $T(G)$ among n -vertex 2-trees G . We will show in the next two sections that they are all the extremal 2-trees. Recall that the n -book B_n is the n -vertex 2-tree obtained from the complete graph K_2 with vertices x and y by iteratively adding simplicial vertices with neighborhood $\{x, y\}$; it has $n - 2$ simplicial vertices.

Theorem 3.2. *The n -vertex 2-tree B_n has $n2^{n-3}$ spanning trees.*

Proof. To count the spanning trees directly, note that those containing the edge xy simply partition the remaining vertices into neighbors of x and neighbors of y ; hence there are 2^{n-2} such trees. When xy is not used, one of the $n - 2$ remaining vertices is adjacent to both x and y , and the remaining $n - 3$ vertices are partitioned into neighbors of x and neighbors of y . Hence $T(B_n) = 2^{n-2} + (n - 2)2^{n-3} = n2^{n-3}$. \square

It is well known that every chordal graph that is not a complete graph has (at least) two nonadjacent simplicial vertices (Dirac [12]).

Lemma 3.3. *An n -vertex 2-tree G has exactly two simplicial vertices if and only if G has a 2-simplicial ordering (v_n, \dots, v_1) such that each v_i is adjacent to v_{i-1} , for $2 \leq i \leq n$. That is, the vertices v_n, \dots, v_1 form a path in order.*

Proof. Since a vertex in a 2-tree with $n \geq 3$ is simplicial if and only if it has degree 2, the neighborhood in G of a simplicial vertex v contains at most one simplicial vertex of $G - v$ (if $n \geq 3$). Hence G has at least as many simplicial vertices as $G - v$.

The only 2-tree with four vertices has exactly two simplicial vertices. If $n \geq 5$, then $G - v$ has the same number of simplicial vertices as G if and only if v is adjacent to some simplicial vertex of $G - v$. Hence in generating a larger 2-tree from the 4-vertex 2-tree by the reverse of a 2-simplicial ordering, the number of simplicial vertices remains 2 if and only if each subsequent added vertex is adjacent to a current simplicial vertex.

When there are exactly two simplicial vertices, this process can be reversed, because any simplicial vertex can be deleted next in a 2-simplicial ordering. In particular, we can *avoid* deleting one of the two original simplicial vertices. Since the number of simplicial vertices cannot decrease below 2, each vertex we delete from the “other end” is adjacent to one simplicial vertex, thus forming the desired path. \square

Various 2-trees having exactly two simplicial vertices. One is the “square” of the n -vertex path P_n , with vertices $\{v_n, \dots, v_1\}$ and edges $\{v_i v_j : |i - j| \leq 2\}$. Another is obtained from P_n by making one endpoint adjacent to all the other vertices. The number of spanning trees in the latter graph has been well studied (for example, see [13]). Surprisingly, all such 2-trees with n -vertices have the same number of spanning trees. We prove a more detailed statement for use later in proving the extremal result.

Definition 3.4. For a set S of edges in a graph G , let $T(G; S)$ denote the number of spanning trees of G containing S ; we write simply $T(G; e)$ when $S = \{e\}$. Also, the Fibonacci sequence $\langle F \rangle$ is defined by $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$. (The solution formula is $F_n = \frac{1}{\sqrt{5}}[\phi^n - (-\phi)^{-n}]$, where $\phi = (1 + \sqrt{5})/2$.)

Theorem 3.5. Let e_0 be an edge in a 2-tree \hat{H} , and let $\alpha = T(\hat{H})$ and $\beta = T(\hat{H}; e_0)$. For $p \geq 1$, let G_p be a 2-tree grown from \hat{H} by successively adding simplicial vertices w_1, \dots, w_p such that for $i \geq 2$ the neighbors of w_i are the endpoints of an edge e_{i-1} incident to w_{i-1} (w_1 is adjacent to the endpoints of e_0). Letting $t_p = T(G_p)$ and $s_p = T(G_p; e_p)$, we have

$$t_p = F_{2p+1}\alpha + F_{2p}\beta \quad \text{and} \quad s_p = F_{2p}\alpha + F_{2p-1}\beta.$$

Proof. We use induction on p . Set $G_0 = \hat{H}$, so $t_0 = \alpha$ and $s_0 = \beta$. Extending the Fibonacci sequence by $F_{-1} = 1$ (preserving the recurrence), the claim holds for $p = 0$.

Consider $p \geq 1$. As in Theorem 2.2, we have $t_p = 2t_{p-1} + s_{p-1}$ and $s_p = t_{p-1} + s_{p-1}$. Using the induction hypothesis,

$$\begin{aligned} t_p &= (2F_{2p-1}\alpha + 2F_{2p-2}\beta) + (F_{2p-2}\alpha + F_{2p-3}\beta) = F_{2p+1}\alpha + F_{2p}\beta, \\ s_p &= (F_{2p-1}\alpha + F_{2p-2}\beta) + (F_{2p-2}\alpha + F_{2p-3}\beta) = F_{2p}\alpha + F_{2p-1}\beta. \end{aligned} \quad \square$$

Corollary 3.6. For an n -vertex 2-tree G with exactly two simplicial vertices, $T(G) = F_{2n-2}$. The value is asymptotic to $\frac{1}{\sqrt{5}}[(1 + \sqrt{5})/2]^{2n-2}$, which is approximately $.1708(2.618)^n$.

Proof. We grow G from the 2-tree with two vertices by $n-2$ steps that add simplicial vertices. Thus we apply Theorem 3.5 with $p = n - 2$ and $\alpha = \beta = 1$ to obtain $T(G) = F_{2n-2}$. \square

4 2-Trees with the Most Spanning Trees

In this section we prove that the n -vertex 2-trees with the most spanning trees are those having exactly two simplicial vertices. We do this by proving that if G is an n -vertex 2-tree with more than two simplicial vertices, then some 2-tree with fewer simplicial vertices than G has more spanning trees than G . We need additional computations based on Theorem 3.5.

Lemma 4.1. *Let H and J be 2-trees properly containing the edge xy . For every set $S \subseteq E(J)$ such that S contains no cycle, $T(H \cup J; S)$ is a strictly increasing function of $T(H; xy)$.*

Proof. We use induction on $|V(J)|$; note $|V(J)| \geq 3$. Let v be a simplicial vertex of J not in $\{x, y\}$, and let $J' = J - v$. Let $N_J(v) = \{w, z\}$, and let $e = wz$ (see Figure 1). For $S \subseteq E(J')$, let $t = T(H \cup J'; S)$ and $s = T(H \cup J'; S \cup \{e\})$. Note that $S \cup \{e\}$ may contain a cycle when $e \notin S$, in which case $s = 0$. Also $S \cup \{e\} = S$ when $e \in S$. By considering which edges in a spanning tree of $H \cup J$ are incident to v (as in earlier arguments), we compute

	if $e \notin S$	if $e \in S$
$T(H \cup J; S)$	$= 2t + s$	$= 2s$
$T(H \cup J; S \cup \{vw\})$	$= t + s$	$= s$
$T(H \cup J; S \cup \{vz\})$	$= t + s$	$= s$
$T(H \cup J; S \cup \{vw, vz\})$	$= s$	$= 0$

When $|V(J)| = 3$, the value t is independent of s and hence constant as a function of s , but adding s gives the desired behavior. When $|V(J)| > 3$ and the specified set is acyclic, by the induction hypothesis all summands are strictly increasing in s . \square

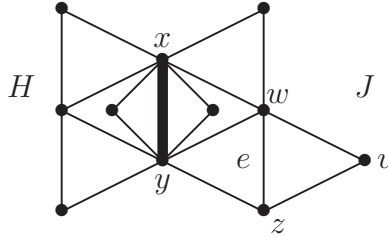


Figure 1: 2-trees H and J sharing an edge xy in Lemma 4.1.

Lemma 4.2. *Let e_0 be an edge in a 2-tree \hat{H} with at least three vertices, and let $\alpha = T(\hat{H})$ and $\beta = T(\hat{H}; e_0)$. For $p \geq 1$, let G_p be a 2-tree grown from \hat{H} by successively adding simplicial vertices w_1, \dots, w_p such that the neighbors of w_i are the endpoints of an edge e_{i-1} incident to w_{i-1} (w_1 is adjacent to the endpoints of e_0). Let e' be the edge of G_1 other than e_1 that is incident to w_1 and e_0 . The numbers of spanning trees containing e_0 and e' satisfy*

$$T(G_p; e_0) = F_{2p+1}\beta \quad \text{and} \quad T(G_p; e') = F_{2p-1}\alpha + F_{2p}\beta$$

In addition, $T(G_p; e_p) > T(G_p; e_0)$ and $T(G_p; e_p) > T(G_p; e')$.

Proof. See Figure 2. By Theorem 3.5, $T(G_p; e_p) = F_{2p+1}\alpha + F_{2p}\beta$. Hence the final statement follows immediately from the claimed formulas for $T(G_p; e_0)$ and $T(G_p; e')$.

For $T(G_p; e_0)$, note that the vertices of e_0 form a separating 2-set in G_p . Let G' be the subgraph of G_p induced by the endpoints of e_0 and the vertices w_1, \dots, w_p . Every spanning tree in G_p containing e_0 is the union of two trees containing e_0 : a spanning tree in \hat{H} and a spanning tree in G' . Note that G' is a 2-tree with two simplicial vertices and can be grown in reverse order by switching the roles of e_0 and e_p . Thus $T(G'; e_0)$ is the value of s_p from Theorem 3.5 when starting with $\alpha = \beta = 1$, since G' is grown from a 2-tree consisting of one edge. Multiplying by $T(G_0; e_0)$ yields $T(G_p; e_0) = (F_{2p} + F_{2p-1})\beta = F_{2p+1}\beta$.

For $T(G_p; e')$, we subtract from $T(G_p)$ the number τ of spanning trees of G_p that do not contain e' . Deleting e' leaves the common vertex of e_0 and e_1 as a cut-vertex. Hence τ is the product of the numbers of spanning trees in the two blocks of $G_p - e'$. One block is just G_0 . The other is a 2-tree with $p + 1$ vertices having exactly two simplicial vertices. By Corollary 3.6, it has F_{2p} spanning trees. By Theorem 3.5, $T(G_p) = F_{2p+1}\alpha + F_{2p}\beta$. Altogether, we have $T(G_p; e') = F_{2p+1}\alpha + F_{2p}\beta - F_{2p}\alpha = F_{2p-1}\alpha + F_{2p}\beta$. \square

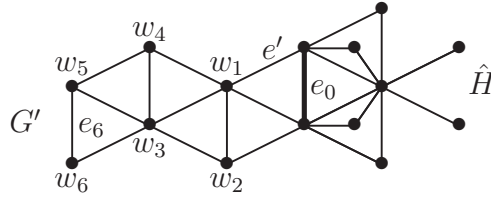


Figure 2: Growing G_6 from \hat{H} containing e_0 in Lemma 4.2.

For an n -vertex 2-tree G with more than two simplicial vertices, an appropriate decomposition of G will allow us to invoke these two lemmas to obtain an n -vertex 2-tree with more spanning trees.

Theorem 4.3. *If $n \geq 4$ and G is an n -vertex 2-tree with more than two simplicial vertices, then G has fewer spanning trees than the n -vertex 2-trees with two simplicial vertices.*

Proof. There is only one 4-vertex 2-tree, and it has two simplicial vertices. For G as specified, we obtain an n -vertex 2-tree G' with more spanning trees. We do this by expressing G as $H \cup J$ with one common edge to apply Lemma 4.1, and we will apply Lemma 4.2 after forming G' by attaching J at an edge belonging to more spanning trees than e in H .

Let v and v' be two simplicial vertices in G ; they are not adjacent. While there remain other simplicial vertices, iteratively delete a simplicial vertex not in $\{v, v'\}$. This begins a 2-simplicial ordering of G . When the process cannot continue further, what remains is a 2-tree H' contained in G ; the only simplicial vertices of H' are v and v' . See Figure 3.

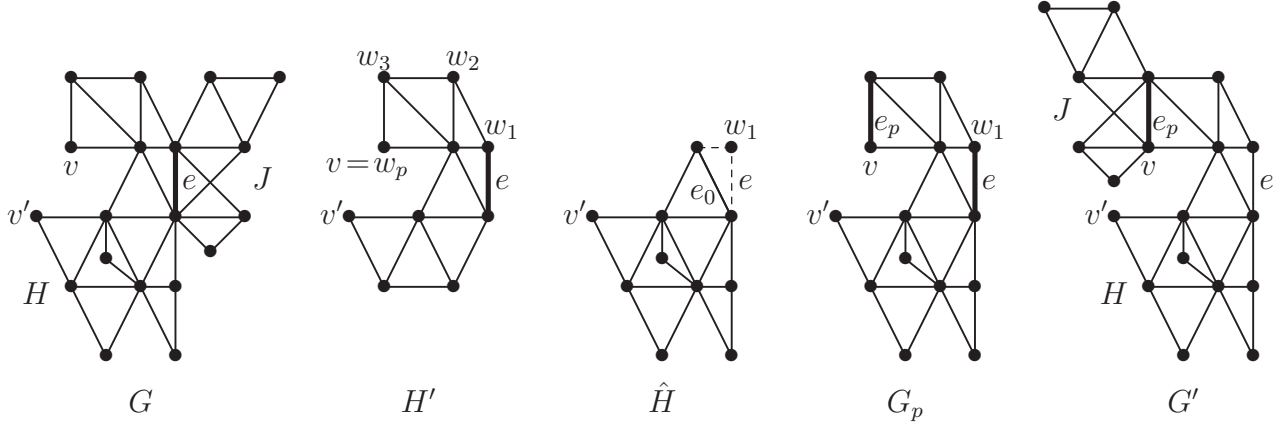


Figure 3: Reattaching J to form G' from G in Theorem 4.3.

Let $q = |V(H')|$. Let (v_q, \dots, v_1) be the 2-simplicial ordering of H' where $v_q = v$ and $v_1 = v'$. Each vertex of G not in H' belongs to a 2-tree that shares exactly one edge with H' and grows from that edge by adding 2-simplicial vertices. Such an edge e cannot be incident to v or v' , since v and v' are simplicial in G . Either e consists of the neighbors of some v_j when v_j is deleted in (v_q, \dots, v_1) , or e consists of v_j and one such neighbor. Let j^* be the largest such index j . Let the resulting edge e be the *crucial edge*, specified as the edge incident to v_{j^*} rather than the one joining the neighbors of v_{j^*} if both choices are available. The example in Figure 3 shows the case where e is incident to v_{j^*} . Let $p = q - j^* + 1$, and let $w_i = v_{j^*-1+i}$ for $1 \leq i \leq p$.

Let \hat{H} be the 2-tree obtained from G by deleting w_p, \dots, w_1 and all the vertices not in H' that belong to the maximal 2-tree sharing only the crucial edge e with H' . In the language of Lemma 4.2, restoring w_p, \dots, w_1 yields a 2-tree G_p with the edge e_p being incident to v and e playing the role of e' or e_0 . By Lemma 4.2, $T(G_p; e_p) > T(G_p; e)$.

Now let $H = G_p$, and let J be the 2-tree induced by the endpoints of e and all the vertices outside H' deleted from G to form \hat{H} ; note that $G = H \cup J$. Let G' be the 2-tree obtained as $H \cup J$ by using e_p as the common edge instead of e . Setting $S = \emptyset$ in Lemma 4.1 now yields $T(G') > T(G)$. \square

5 2-Trees with the Fewest Spanning Trees

We close by proving that the unique n -vertex 2-tree having the fewest spanning trees is B_n , the 2-tree with $n - 2$ simplicial vertices (see Theorem 3.2).

Theorem 5.1. *If G is a n -vertex 2-tree other than B_n , then $T(G) > T(B_n)$.*

Proof. It suffices to show that some n -vertex 2-tree other than G has fewer spanning trees than G . The book B_n is characterized by the fact that every two simplicial vertices have

the same neighborhood. If $G \neq B_n$, then let v_1 and v_2 be simplicial vertices in G whose neighborhoods are not equal. Let $H = G - \{v_1, v_2\}$.

For $i \in \{1, 2\}$, let e_i be the edge joining the neighbors of v_i , let $\beta_i = T(H; e_i)$, and let G_i be the graph obtained from H by adding two simplicial vertices each having neighborhood $N_G(v_i)$ (see Figure 5.1). Let $\gamma = T(H; \{e_1, e_2\})$.

Spanning trees in G have two, three or four edges incident to $\{v_1, v_2\}$; we compute $T(G) = 4T(H) + 2\beta_1 + 2\beta_2 + \gamma$. Spanning trees in G_i have two or three edges not in H ; we compute $T(G_i) = 4T(H) + 4\beta_i$. Thus

$$\begin{aligned} 2T(G) &= 8T(H) + 4\beta_1 + 4\beta_2 + 2\gamma \\ T(G_1) + T(G_2) &= 8T(H) + 4\beta_1 + 4\beta_2. \end{aligned}$$

Since a spanning tree can be grown from any subtree, $\gamma \neq 0$. Thus $T(G_1) + T(G_2) < 2T(G)$, and G_1 or G_2 has fewer spanning trees than G . \square

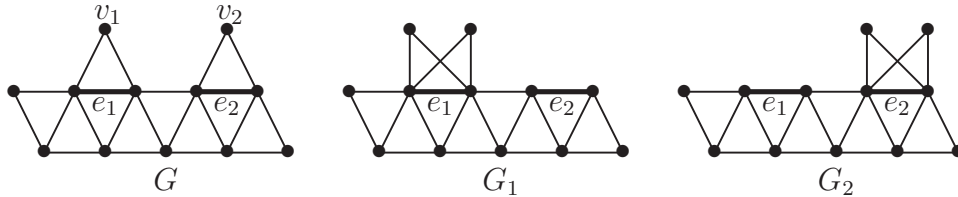


Figure 4: Graphs G_1 and G_2 compared with G in Theorem 5.1.

References

- [1] H N Gabow, E W Myers: Finding all spanning trees of directed and undirected graphs. SIAM Journal on Computing **7**(3) 280–287 (1978)
- [2] S Kapoor, H Ramesh: Algorithms for enumerating all spanning trees of undirected and weighted graphs. SIAM Journal on Computing **24**(2) 247–265 (1995)
- [3] S Kapoor, H Ramesh: An algorithm for enumerating all spanning trees of a directed graph. Algorithmica **27** 120–130 (2000)
- [4] G Kirchhoff: Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird. Ann. Phys. Chem. **72** 497–508 (1847)
- [5] T Matsui: A flexible algorithm for generating all the spanning trees in undirected graphs. Algorithmica **18** 530–543 (1997)

- [6] R C Read, R E Tarjan: Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks* 237–252 (1975)
- [7] A Shioura, A Tamura: Efficiently scanning all spanning trees of an undirected graph. *J. Operation Research Society Japan* **38** 331–344 (1993)
- [8] Akiyoshi Shioura, Akihisa Tamura, Takeaki Uno: An optimal algorithm for scanning all spanning trees of undirected graphs. *SIAM Journal on Computing* **26**(3) 678–692 (1997)
- [9] Weigen Yan, Fuji Zhang: Enumeration of spanning trees of graphs with rotational symmetry. *Journal of Combinatorial Theory, Series A* **118**(4) 1270 – 1290 (2011)
- [10] A Cayley: A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics* 376–378 (1889)
- [11] M C Golumbic: *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York (1980)
- [12] G A Dirac: On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg* **25**(1) 71–76 (1961)
- [13] A J W Hilton: The number of spanning trees of labelled wheels, fans and baskets. *The Proc. of the Oxford Conf. on Comb.* 203–206 (1972)